EBook Gratuito

APPRENDIMENTO testng

Free unaffiliated eBook created from
Stack Overflow contributors.

#testng

Sommario

Di

Capitolo 1: Iniziare con testng

Osservazioni

Questa sezione fornisce una panoramica di cosa è il testng e perché uno sviluppatore potrebbe volerlo usare.

Dovrebbe anche menzionare eventuali soggetti di grandi dimensioni all'interno di testng e collegarsi agli argomenti correlati. Poiché la Documentazione per testng è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data
1.0	2017/06/07

Examples

Installazione o configurazione

TestNG richiede JDK 7 o superiore per l'uso.

Secondo http://testng.org/doc/download.html per installare testng è necessario aggiungere la dipendenza testng al file maven pom.xml o gradle build.gradle

Maven:

```
<repositories>
  <repository>
    <id>jcenter</id>
    <name>bintray</name>
    <url>http://jcenter.bintray.com</url>
  </repository>
</repositories>

<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>6.9.12</version>
  <scope>test</scope>
</dependency>
```

Gradle:

```
repositories {
    jcenter()
```

```
}

dependencies {
    testCompile 'org.testng:testng:6.9.12'
}
```

Altre opzioni possono essere trovate nella pagina ufficiale .

Programma rapido con TestNG

```
package example;

import org.testng.annotations.*; // using TestNG annotations

public class Test {

 @BeforeClass
 public void setUp() {
   // code that will be invoked when this test is instantiated
 }

 @Test(groups = { "fast" })
 public void aFastTest() {
   System.out.println("Fast test");
 }

 @Test(groups = { "slow" })
 public void aSlowTest() {
    System.out.println("Slow test");
 }

}
```

Il metodo `setUp()` verrà richiamato dopo che la classe di test è stata `setUp()` e prima che venga eseguito qualsiasi metodo di prova. In questo esempio, ci sarà la gestione dei gruppi veloce, così `aFastTest()` verrà richiamato mentre `aSlowTest()` sarà saltato.

TestNG Hello World Example

Scrivere ed eseguire un semplice programma `TestNG` è principalmente un processo in 3 fasi.

1. Codice: scrivere la business logic del test e annotarlo con le annotazioni TestNG
2. Configura: aggiungi le informazioni del test in `testng.xml` o in `build.xml`
3. Esegui TestNG - può essere richiamato dalla riga di comando, ANT, IDE come Eclipse, IntelliJ's IDEA)

Breve spiegazione dell'esempio (cosa è necessario testare) :

Abbiamo una classe `RandomNumberGenerator` che ha un metodo `generateFourDigitPin` che genera un PIN di 4 cifre e restituisce come `int` . Quindi qui vogliamo testare se quel numero casuale è se di 4 cifre o meno. Di seguito è riportato il codice:

Classe da testare :

```
package example.helloworld;

public class RandomNumberGenerator {

public int generateFourDigitPin(){
    return (int)(Math.random() * 10000);
}
}
```

La classe di test di TestNG :

```
package example.helloworld;

import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class TestRandomNumberGenerator {

    RandomNumberGenerator rng = null;

    @BeforeClass
    public void deSetup(){
        rng = new RandomNumberGenerator();
    }

    @Test
    public void testGenerateFourDigitPin(){
        int randomNumber = rng.generateFourDigitPin();
        Assert.assertEquals(4, String.valueOf(randomNumber).length());
    }

    @AfterClass
    public void doCleanup(){
        //cleanup stuff goes here
    }
}
```

Ther testng.xml :

```
<suite name="Hello World">
    <test name="Random Number Generator Test">
        <classes>
            <class name="example.helloworld.TestRandomNumberGenerator" />
        </classes>
    </test>
</suite>
```

Esegui la suite TestNG con Gradle

Esempio di file `build.gradle` :

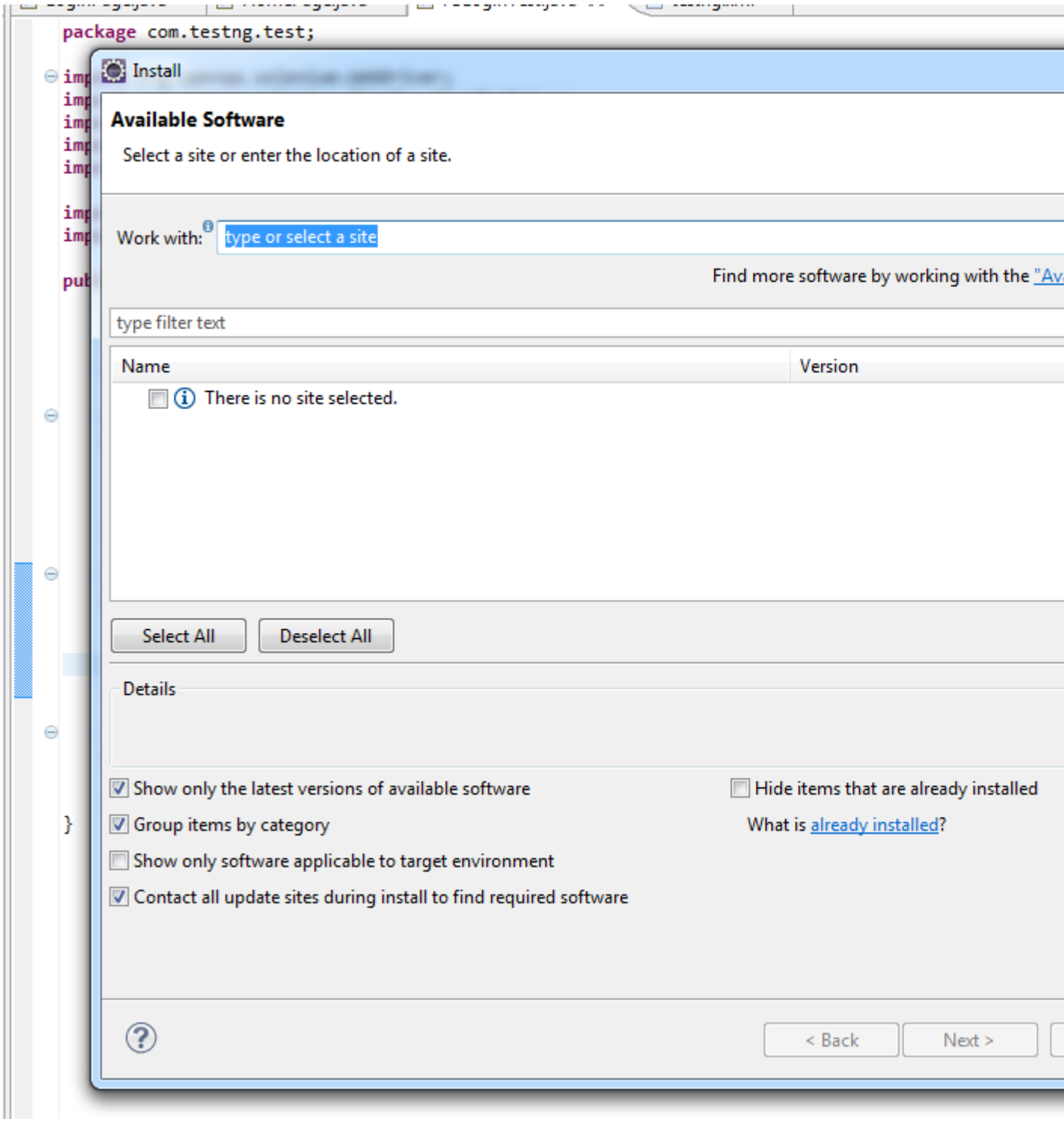
```
plugin: 'java'

repositories {
```

```
    mavenLocal()
    mavenCentral()
    jcenter()
}

dependencies {
    compile "org.testng:testng:6.9.12"
}

test {
    useTestNG() {
    suiteXmlBuilder().suite(name: 'Sample Suite') {
        test(name : 'Sample Test') {
            classes('') {
                'class'(name: 'your.sample.TestClass')
            }
        }
    }
}
```

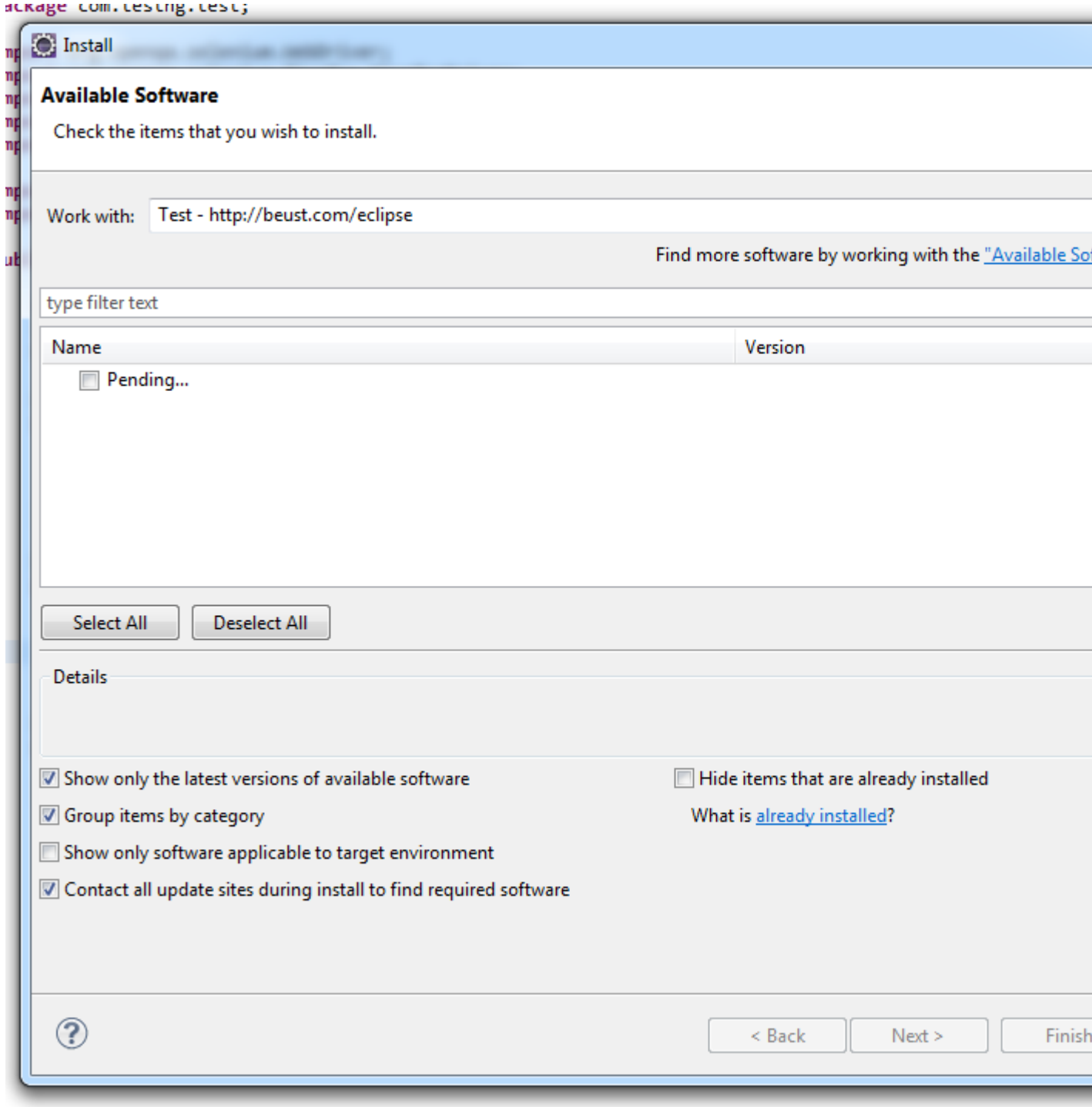
Come configurare TestNG in Eclipse ed eseguire test usando xml

Come installare TestNG in eclissi

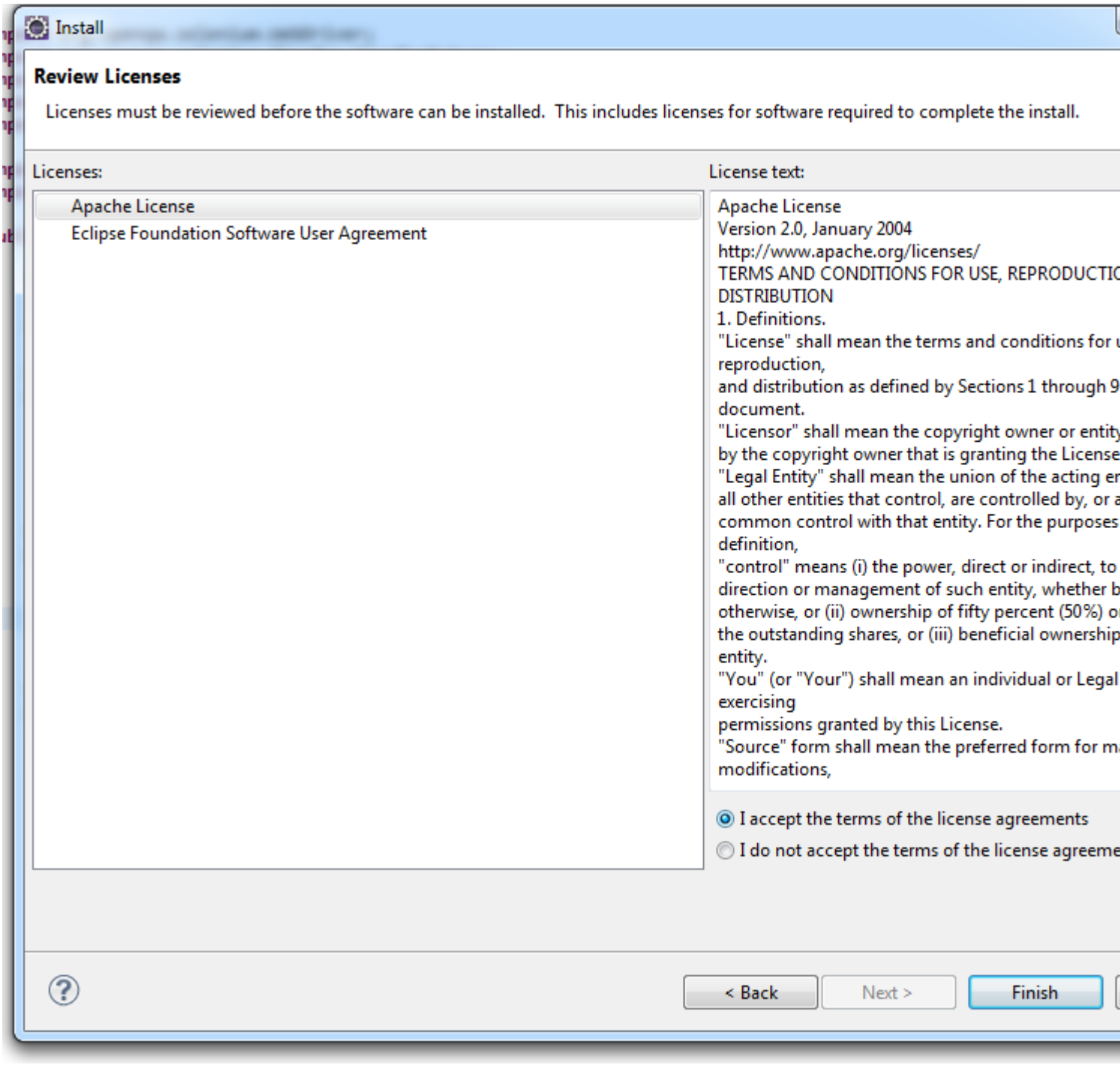
1. Apri eclissi
2. Fare clic su Guida> Installa nuovo software



3. Fai clic su Aggiungi
4. Fornisci nome e URL - http://beust.com/eclipse



5. Seleziona TestNG
6. Fare clic su Avanti



7. Fai clic su Fine
8. Ci vorrà del tempo per installare TestNG

Una volta installato, riavviare eclipse.

Consente di creare un progetto TestNG

1. File> Nuovo> Progetto Java> Fornire un nome e fare clic su Fine

2. Crea una classe come TestNGClass

3. Crea la seguente lezione

    1.LoginPage.class

2.HomePage.class

3.FBLoginTest.class

Ecco il codice:

Classe LoginPage

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class LoginPage {

    @FindBy(id = "email")
    private WebElement username;

    @FindBy(id = "pass")
    private WebElement password;

    @FindBy(xpath = ".//input[@data-testid='royal_login_button']")
    private WebElement login;

    WebDriver driver;

    public LoginPage(WebDriver driver){
        this.driver = driver;
         PageFactory.initElements(driver, this);
    }
    public void enterUserName(String name){
        username.clear();
        username.sendKeys(name);
    }

    public void enterPassword(String passwrd){
        password.clear();
        password.sendKeys(passwrd);
    }


    public HomePage clickLoginButton(){
        login.click();
        return new HomePage(driver);
    }
}
```

Classe HomePage .

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

public class HomePage {

    @FindBy(id = "userNavigationLabel")
    private WebElement userDropdown;
```

```
    WebDriver driver;

    public HomePage(WebDriver driver){
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    public boolean isUserLoggedIn(){
        return userDropdown.isDisplayed();
    }

}
```

Classe FBLoginTest

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.AfterClass;

import com.testng.pages.HomePage;
import com.testng.pages.LoginPage;

public class FBLoginTest {

    WebDriver driver;
    LoginPage loginPage;
    HomePage homePage;

    @BeforeClass
    public void openFBPage(){
        driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        loginPage = new LoginPage(driver);
    }

    @Test
    public void loginToFB(){
        loginPage.enterUserName("");
        loginPage.enterPassword("");
        homePage = loginPage.clickLoginButton();
        Assert.assertTrue(homePage.isUserLoggedIn());
    }

    @AfterClass
    public void closeBrowser(){
        driver.quit();
    }

}
```

Ecco il test xml: tasto destro del mouse su Progetto creare un file xml e copiare incollare questo contenuto.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<suite name="Suite">
  <test name="Test">
    <classes>
      <class name="com.testng.FBLoginTest"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

Come aggiungere il vaso autonomo di selenio:
Scarica l'ultimo jar autonomo di selenio e aggiungilo nel percorso di creazione del progetto.

1. Fare clic con il tasto destro su Progetto> Crea percorso> Configura percorso build> Seleziona librerie> Aggiungi giare esterne

Come eseguire TestNG xml? Fare clic con il tasto destro su xml> Esegui come> TestNGSuite

Happy Coding :)

Leggi Iniziare con testng online: https://riptutorial.com/it/testng/topic/5393/iniziare-con-testng

Capitolo 2: @Test Annotation

Sintassi

- @Test
- @Test (attribute1 = attributeValue, atrribute2 = attributeValue, ecc)

Parametri

Parametro	Dettagli
AlwaysRun	Se impostato su true, questo metodo di test verrà sempre eseguito anche se dipende da un metodo non riuscito.
dataProvider	Il nome del fornitore di dati per questo metodo di prova.
dataProviderClass	La classe in cui cercare il fornitore di dati. Se non specificato, il fornitore di dati verrà considerato sulla classe del metodo di test corrente o su una delle sue classi di base. Se questo attributo è specificato, il metodo del fornitore di dati deve essere statico sulla classe specificata.
dependsOnGroups	L'elenco dei gruppi dipende da questo metodo.
dependsOnMethods	L'elenco di metodi da cui dipende questo metodo.
descrizione	La descrizione per questo metodo.
abilitato	Se i metodi su questa classe / metodo sono abilitati.
expectedExceptions	L'elenco di eccezioni che un metodo di prova dovrebbe generare. Se non viene lanciata alcuna eccezione o una diversa da questa lista, questo test verrà contrassegnato come non riuscito.
gruppi	L'elenco dei gruppi a cui appartiene questa classe / metodo.
invocationCount	Il numero di volte che questo metodo deve essere invocato.
invocationTimeOut	Il numero massimo di millisecondi che questo test dovrebbe richiedere per il tempo cumulativo di tutti gli account di chiamata. Questo attributo verrà ignorato se invocationCount non è specificato.
priorità	La priorità per questo metodo di prova. Le priorità più basse saranno programmate per prime.
successPercentage	La percentuale di successo prevista da questo metodo

Parametro	Dettagli
singleThreaded	Se impostato su true, tutti i metodi su questa classe di test sono garantiti per l'esecuzione nello stesso thread, anche se i test vengono attualmente eseguiti con `parallel="methods"` . Questo attributo può essere utilizzato solo a livello di classe e verrà ignorato se utilizzato a livello di metodo. **Nota** : questo attributo si chiamava sequenziale (ora deprecato).
tempo scaduto	Il numero massimo di millisecondi che questo test dovrebbe richiedere.
threadPoolSize	La dimensione del pool di thread per questo metodo. Il metodo verrà richiamato da più thread come specificato da invocationCount. **Nota** : questo attributo viene ignorato se invocationCount non è specificato

Examples

Esempio rapido sull'annotazione @Test

`@Test` annotazione `@Test` può essere applicata a qualsiasi **classe** o **metodo** . Questa annotazione contrassegna una classe o un metodo come parte del test.

1. `@Test` al livello del metodo - contrassegna il metodo annotato come metodo di prova
2. `@Test` a livello di classe
    - L'effetto di un'annotazione `@Test` livello di classe consiste nel rendere tutti i metodi pubblici della classe come metodi di test anche se non sono annotati.
    - `@Test` annotazione di `@Test` può anche essere ripetuta su un metodo se si desidera aggiungere determinati attributi.

Esempio di `@Test` al livello del metodo :

```
import org.testng.annotations.Test;

public class TestClass1 {
    public void notTestMethod() {
    }

    @Test
    public void testMethod() {
    }
}
```

Esempio di `@Test` a livello di classe :

```
import org.testng.annotations.Test;

@Test
public class TestClass2 {
    public void testMethod1() {
```

```
    }

    @Test
    public void testMethod2() {
    }
}
```

Leggi @Test Annotation online: https://riptutorial.com/it/testng/topic/6716/-test-annotation

Capitolo 3: Gruppi TestNG

Sintassi

- @Test (groups = {"group1", "group.regression"}, dependsOnGroups = {"group2", "group3"})

Examples

Configurazione Gruppi TestNG ed esempio di base

I gruppi possono essere configurati sotto l'elemento `Suite` e / o `Test` di `testng.xml` . Tutti i gruppi contrassegnati come inclusi in `tesng.xml` verranno considerati per l'esecuzione, escluso uno verrà ignorato. Se un `@Test` metodo ha più gruppi e da questi gruppi, se uno stesso gruppo è escluso in `testng.xml` che `@Test` metodo non verrà eseguito.

Di seguito è riportata la tipica configurazione `testng.xml` a livello di `Test` per i gruppi in esecuzione:

```
<suite name="Suite World">
<test name="Test Name">
  <groups>
    <run>
      <include name="functest" />
      <exclude name="regtest" />
    </run>
  </groups>
  <classes>
    <class name="example.group.GroupTest"/>
  </classes>
</test>
</suite>
```

Ecco come sarà la classe di test:

```
package example.group;

import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class GroupTest {

    @BeforeClass
    public void deSetup(){
        //do configuration stuff here
    }

    @Test(groups = { "functest", "regtest" })
    public void testMethod1() {
    }

    @Test(groups = {"functest", "regtest"} )
    public void testMethod2() {
```

```
    }

    @Test(groups = { "functest" })
    public void testMethod3() {
    }

    @AfterClass
    public void cleanUp(){
        //do resource release and cleanup stuff here
    }
}
```

Sull'esecuzione `GroupTest TestNG` classe solo `testMethod3()` verrà eseguita.

Spiegazione:

- `<include name="functest" />` tutti i metodi di test del gruppo `functest` sono idonei per l'esecuzione se non esclusi da qualsiasi altro gruppo.
- `<exclude name="regtest" />` nessun metodo di test del gruppo `regtest` è idoneo per l'esecuzione.
- `testMethod1()` e `testMethod2()` sono nel gruppo `regtest` , quindi non avranno eseguito.
- `testMethod3()` è nel gruppo `regtest` , quindi verrà eseguito.

TestNG MetaGroups - Gruppi di gruppi

TestNG consente di definire gruppi che possono includere altri gruppi. I MetaGroup combinano logicamente uno o più gruppi e controllano l'esecuzione dei metodi `@Test` appartenenti a quei gruppi.

Nell'esempio sottostante ci sono vari metodi `@Test` che appartengono a diversi gruppi. Pochi sono specifici per stack particolari e pochi sono regressione e test di accettazione. Qui possono essere creati i MetaGroup. **Selezioniamo** due semplici **MetaGroup** :

1. `allstack` - include entrambi i gruppi `liux.jboss.oracle` e `aix.was.db2` e consente a tutti i metodi di test appartenenti a uno di questi gruppi di funzionare insieme.
2. `systemtest` - include i gruppi `allstack` , `regression` e `acceptance` e consente a tutti i metodi di test appartenenti a uno di questi gruppi di funzionare insieme.

configurazione **testng.xml**

```
<suite name="Groups of Groups">
    <test name="MetaGroups Test">
        <groups>
            <!-- allstack group includes both liux.jboss.oracle and aix.was.db2 groups -->
            <define name="allstack">
                <include name="liux.jboss.oracle" />
                <include name="aix.was.db2" />
            </define>

            <!-- systemtest group includes all groups allstack, regression and acceptance -->
            <define name="systemtest">
                <include name="allstack" />
                <include name="regression" />
```

```
                <include name="acceptance" />
            </define>

            <run>
                <include name="systemtest" />
            </run>
        </groups>

        <classes>
            <class name="example.group.MetaGroupsTest" />
        </classes>
    </test>

</suite>
```

MetaGroupsTest class

```
package example.group;

import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

public class MetaGroupsTest {

    @BeforeMethod
    public void beforeMethod(){
        //before method stuffs - setup
    }

    @Test(groups = { "liux.jboss.oracle", "acceptance" })
    public void testOnLinuxJbossOracleStack() {
        //your test logic goes here
    }

    @Test(groups = {"aix.was.db2", "regression"} )
    public void testOnAixWasDb2Stack() {
        //your test logic goes here
    }

    @Test(groups = "acceptance")
    public void testAcceptance() {
        //your test logic goes here
    }

    @Test(groups = "regression")
    public void testRegression(){
        //your test logic goes here
    }

    @AfterMethod
    public void afterMthod(){
        //after method stuffs - cleanup
    }
}
```

Leggi Gruppi TestNG online: https://riptutorial.com/it/testng/topic/5821/gruppi-testng

Capitolo 4: Test parametrizzati

Examples

Fornitori di dati

I fornitori di dati consentono di creare più input di test da eseguire all'interno di un test. Consideriamo un test che verifica che i numeri siano raddoppiati correttamente. Per creare un fornitore di dati, fornire un metodo statico che restituisce `Object[][]` o `Iterator<Object[]>` (quest'ultimo consente il calcolo lazy degli input di test) annotato con `@DataProvider` annotazione `@DataProvider` , con il `name` proprietà come stringa univoca che identifica il fornitore.

```
import org.testng.annotations.DataProvider;

public class DoublingDataProvider {
    public final static String DOUBLING_DATA_PROVIDER = "doublingDataProvider";

    @DataProvider(name = DOUBLING_DATA_PROVIDER)
    public static Object[][] doubling() {
        return new Object[][]{
                new Object[]{1, 2},
                new Object[]{2, 4},
                new Object[]{3, 6}
        };
    }
}
```

Nel caso precedente ciascun `Object[]` rappresenta un insieme di dati per un singolo caso di test: qui il numero da raddoppiare, seguito dal valore previsto dopo il raddoppio.

Per utilizzare il fornitore di dati, riempire la proprietà `dataProvider` del test con il nome del provider. Se il metodo del provider è stato definito al di fuori della classe di test o delle sue classi di base, è necessario specificare anche la proprietà `dataProviderClass` . Il metodo di test dovrebbe prendere i parametri corrispondenti agli elementi della descrizione del caso di test - qui sono due inte.

```
import org.testng.annotations.Test;

import static org.testng.Assert.assertEquals;

public class DoublingTest {

    @Test(dataProvider = DoublingDataProvider.DOUBLING_DATA_PROVIDER, dataProviderClass =
DoublingDataProvider.class)
    public void testDoubling(int number, int expectedResult) {
        assertEquals(number * 2, expectedResult);
    }
}
```

Leggi Test parametrizzati online: https://riptutorial.com/it/testng/topic/5684/test-parametrizzati

Capitolo 5: TestNG - Procedura di esecuzione

Examples

Procedura di esecuzione dei metodi dell'API di test di TestNG

```
public class TestngAnnotation {
   // test case 1
   @Test
   public void testCase1() {
      System.out.println("in test case 1");
   }

   // test case 2
   @Test
   public void testCase2() {
      System.out.println("in test case 2");
   }

   @BeforeMethod
   public void beforeMethod() {
      System.out.println("in beforeMethod");
   }

   @AfterMethod
   public void afterMethod() {
      System.out.println("in afterMethod");
   }

   @BeforeClass
   public void beforeClass() {
      System.out.println("in beforeClass");
   }

   @AfterClass
   public void afterClass() {
      System.out.println("in afterClass");
   }

   @BeforeTest
   public void beforeTest() {
      System.out.println("in beforeTest");
   }

   @AfterTest
   public void afterTest() {
      System.out.println("in afterTest");
   }

   @BeforeSuite
   public void beforeSuite() {
      System.out.println("in beforeSuite");
   }

   @AfterSuite
   public void afterSuite() {
      System.out.println("in afterSuite");
```

```
   }

}
```

creiamo il file testng.xml in C:> WORKSPACE per eseguire le annotazioni.

```
<suite name="Suite1">
  <test name="test1">
    <classes>
       <class name="TestngAnnotation"/>
    </classes>
  </test>
</suite>
```

C: \ WORKSPACE> javac TestngAnnotation.java

Ora esegui testng.xml, che eseguirà il test case definito nella classe Test Case fornita.

```
in beforeSuite
in beforeTest
in beforeClass
in beforeMethod
in test case 1
in afterMethod
in beforeMethod
in test case 2
in afterMethod
in afterClass
in afterTest
in afterSuite

===============================================
Suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

La procedura di esecuzione è la seguente:

1. Prima di tutto, il metodo **beforeSuite ()** viene eseguito solo una volta.
2. Infine, il metodo **afterSuite ()** viene eseguito solo una volta.
3. Anche i metodi **beforeTest ()** , **beforeClass ()** , **afterClass ()** e **afterTest ()** vengono eseguiti solo una volta.
4. **Il** metodo **beforeMethod ()** viene eseguito per ogni caso di test ma prima dell'esecuzione del test case.
5. **Il** metodo **afterMethod ()** viene eseguito per ogni caso di test ma dopo l'esecuzione del test case.
6. Tra **beforeMethod ()** e **afterMethod ()** , ogni caso di test viene eseguito.

Leggi TestNG - Procedura di esecuzione online: https://riptutorial.com/it/testng/topic/7889/testng---procedura-di-esecuzione

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con testng	Atul Dwivedi, Community, Idos, mackowski, RocketRaccoon, Sudha Velan
2	@Test Annotation	Atul Dwivedi, Benoit
3	Gruppi TestNG	Atul Dwivedi
4	Test parametrizzati	Benoit, mszymborski
5	TestNG - Procedura di esecuzione	Shrikant